FUNDAMENTOS DE PROGRAMACIÓN

Práctica de Laboratorio: Creación de un proyecto básico en Java

Autores: Toñi Reina, Mariano González. **Última modificación:** 17/2/2022

Objetivos:

- Conocer cuándo un tipo se implementa con una clase y cuándo se puede implementar con un record.
- Distinguir entre propiedades básicas y propiedades derivadas.
- Saber crear una clase, y
  - Definir los atributos
  - Implementar los constructores
  - Implementar los getters y setters, y otras operaciones
  - Implementar el método toString
- Saber crear un record, y
  - Definir los atributos
  - Implementar los constructores
  - Implementar los getters
  - Implementar el método toString

Para realizar la práctica, sigue los pasos que te indicamos a continuación:

1. Construye el proyecto

Ejecuta Eclipse y construye un proyecto Java de nombre `L01-Tipos` con la siguiente estructura. Para cada tipo a implementar, crea los dos paquetes que se indican en la descripción del tipo: uno en el que se recoge el tipo a implementar y los tipos auxiliares para implementar el tipo, y otro para crear las clases de test para probar la funcionalidad implementada en el tipo.

2. Implementa el tipo

1. Determina si el tipo se puede implementar con un record o con una clase.
2. Crea el tipo en el paquete indicado.
3. Si el tipo tiene alguna propiedad de tipo enumerado, crea el tipo enumerado.
4. Si el tipo necesita un tipo auxiliar (otra clase o record), impleméntalo en el paquete fp.tipos.common.
5. Para el tipo:
   a. Declara los atributos utilizando los nombres y tipos de datos que se proporcionan en la descripción.
   b. Escribe los métodos constructores indicados en la descripción.
   c. Escribe los métodos consultores (*getter*) necesarios, según la descripción del tipo.
   d. Escribe los métodos modificadores (*setter*) necesarios, según la descripción del tipo.
   e. Escribe automáticamente el método *toString*, según la descripción del tipo.

3. Prueba el tipo

Escribe en el paquete de test correspondiente una clase de test (por ejemplo, `TestCancion`) y asegúrate de marcar la casilla para que se cree en ella un método `main`. Escribe en este método instrucciones para crear objetos del tipo usando todos sus constructores, modificar los atributos que sean modificables, y mostrarlos en la pantalla. Realiza diferentes pruebas.

Descripción de los tipos

1. Cancion

Paquetes: fp.tipos.musica, fp.tipos.musica.test

Propiedades:
- *titulo*, de tipo String, consultable y modificable.
- *artista*, de tipo String, consultable y modificable. Representa al intérprete de la canción.
- *duracion*, de tipo Duration, consultable y modificable.
- *fechaLanzamiento*, de tipo LocalDate, consultable y modificable.
- *genero*, de tipo Genero, consultable y modificable. Puede tomar los valores: POP, ROCK, FOLK.
- *formatoCorto,* de tipo String, consultable. Cadena que representa una canción con el siguiente formato: el título de la canción, seguido del artista entre paréntesis y la duración, por ejemplo, "Whole Lotta Love (Led Zeppelin) 3:20"

Constructores:
- C1: recibe como parámetros el título y el artista y crea una canción con duración de 0 segundos, y el resto de atributos nulos.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

2. Pelicula

Paquetes: fp.tipos.cine, fp.tipos.cine.test

Propiedades:
- *titulo,* de tipo String, consultable. Representa el título de la película en su versión española (puede coincidir con el título original).
- *fecha de estreno,* de tipo LocalDate, consultable y modificable. Fecha en la que se estrenó la película.
- *duración,* de tipo Integer, consultable y modificable. Representa la duración (en minutos) de la película.
- *tipo de metraje,* de tipo TipoMetraje, consultable. Se calcula a partir de la duración. Los tipos de metraje posibles son CORTOMETRAJE (menos de 30 minutos), MEDIOMETRAJE (de 30 a 60 minutos) y LARGOMETRAJE (más de 60 minutos).
- *formatoCorto,* de tipo String. Cadena que contiene el título, seguido del año de la fecha de estreno entre paréntesis, siempre que esté disponible la fecha de estreno (por ejemplo, "Tiempos modernos (1936)". Si no está disponible la fecha de estreno, entonces se mostrará solo el título (por ejemplo, "Un monstruo viene a verme").

Constructores:
- C1: recibe un parámetro por cada propiedad básica del tipo.
- C2: recibe sólo el título (el resto de las propiedades básicas se quedan con el valor null).

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

3. PartidoFutbol

Paquetes: fp.tipos.futbol, fp.tipos.futbol.test

Propiedades:
- *fecha*, de tipo LocalDateTime, consultable. Fecha de celebración y hora de comienzo del partido.
- *equipo local*, de tipo String, consultable. Nombre del equipo local.
- *equipo visitante*, de tipo String, consultable. Nombre del equipo visitante.
- *goles local*, de tipo Integer, consultable. Número de goles del equipo local.
- *goles visitante*, de tipo Integer, consultable. Número de goles del equipo visitante.

- *resultado*, de tipo ResultadoQuiniela, consultable. Puede tomar los valores UNO, EQUIS, DOS. El resultado será UNO, si el equipo local tiene más goles que el equipo visitante; será EQUIS, si los dos equipos han marcado el mismo número de goles, y DOS, si el equipo visitante ha marcado más goles que el local.
- *cadena con formato*, de tipo String. Devuelve una cadena con la fecha del partido, seguido del equipo local, el equipo visitante, los goles del equipo local, los goles del equipo visitante, y entre paréntesis el resultado de la quiniela, tal y como se muestra en el siguiente ejemplo: "24-09-16-> Sporting Gijón vs Barcelona: 0-5 (2)". Tenga en cuenta que el resultado de la quiniela se debe representar con los caracteres 1, X ó 2.

Constructores:

- C1: recibe un parámetro por cada propiedad básica del tipo.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

4. Hotel:

Paquetes: fp.tipos.hoteles, fp.tipos.hoteles.test

Propiedades:

- *nombre*, de tipo String, consultable. Nombre del hotel.
- *dirección*, de tipo String, consultable. Dirección del hotel.
- *ciudad*, de tipo String, consultable. Ciudad en la que está situada el hotel.
- *telefono*, de tipo String, consultable. Número de teléfono del hotel.
- *cadena hotelera*, de tipo String, consultable. Nombre de la cadena hotelera a la que pertenece el hotel.
- *descripción*, de tipo String, consultable. Contiene una pequeña descripción del hotel.
- *categoría del hotel*, de tipo CategoriaHotelera, consultable. Puede tomar los valores UNA, DOS, TRES, CUATRO O CINCO, que representan el número de estrellas del hotel. También se incluirá el valor OTROS, para aquellos tipos de alojamiento en los que no proceda hablar de la categoría.
- *tipo de alojamiento*, de tipo TipoAlojamiento, consultable. Puede tomar los valores APARTAMENTO, HOTEL, PENSION, BED_AND_BREAKFAST.
- *categoría de precio*, de tipo CategoriaPrecio, consultable. Puede tomar los valores BAJA, MEDIA, ALTA y LUJO, que categoriza los hoteles según el precio de las habitaciones.
- *puntuación*, de tipo Float, consultable. Media de las puntuaciones que le otorgan los usuarios del sitio web.
- *número de comentarios*, de tipo Integer, consultable. Número de usuarios que han realizado un comentario sobre el hotel y le han dado una puntuación.
- *admite mascotas*, de tipo Boolean, consultable. Indica si el hotel admite mascotas.
- *está adaptado*, de tipo Boolean, consultable. Indica si el hotel está adaptado para minusválidos.
- *coordenadas,* de tipo Coordenada, consultable. Coordenadas en las que está situado el hotel.
- *cadena con formato,* de tipo String, consultable. El nombre del hotel, seguido entre paréntesis de tantos asteriscos como estrellas tenga (por ejemplo, "Hotel Alfonso XIII - A Luxury Collection Hotel (*****)"), o N/A si la categoría hotelera es OTROS (ej., "Hotel Dawson Place, Juliette's Bed and Breakfast (N/A)").

Constructores:

- C1: recibe un parámetro por cada propiedad básica del tipo.
- C2: recibe el nombre, la dirección, la ciudad, el teléfono, la cadena hotelera, y la categoría, y permite crear un hotel de precio medio, sin ningún comentario de los usuarios, que no admite mascotas ni está adaptado.
- C3: recibe el nombre, la cadena hotelera y la categoría del hotel, y permite crear un hotel de precio medio, sin comentarios, que no admite mascotas ni está adaptado (el resto de las propiedades básicas quedarán con el valor null).

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

Tipo auxiliar Coordenada

Propiedades:
- *latitud*, de tipo Double, consultable. Latitud.
- *longitud*, de tipo Double, consultable. Longitud.

Constructores:
- C1: recibe un parámetro por cada propiedad básica del tipo.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

Otras operaciones:
- Double getDistancia(Coordenada c): devuelve la distancia euclídea entre la coordenada con la que se invoca al método y la coordenada c.
- Double getDistanciaHaversine(Coordenada c): devuelve la distancia de Haversine entre la coordenada con la que se invoca al método y la coordenada c.

Para calcular la distancia de Haversine, aplica la siguiente fórmula: si c1 y c2 son dos coordenadas, c1 tiene lat1 y lon1 como latitud y longitud, y c2 tiene lat2 y lon2 como latitud y longitud, entonces la distancia de Haversine se calcula de la siguiente forma:

1. R = radio de la Tierra (define una constante con valor 6371.0).
2. Δlat = lat2− lat1, diferencia de latitudes, en radianes (usa Math.toRadians para convertir a radianes).
3. Δlong = long2− long1, diferencia de longitudes, en radianes.
4. a = $\sin^2(\Delta lat/2) + \cos(lat1) \cdot \cos(lat2) \cdot \sin^2(\Delta long/2)$, donde tanto lat1 como lat2 deben estar en radianes (usa los métodos Math.pow, Math.sin y Math.cos para implementar el cálculo).
5. c = 2 · atan2(√a, √(1−a)) (usa Math.atan2 y Math.sqrt).
6. distancia Haversine = R · c

5. Vuelo

Paquetes: fp.tipos.aeropuerto, fp.tipos.aeropuerto.test

Propiedades:
- *codigo*, de tipo String, consultable. Código del vuelo.
- *origen*, de tipo String, consultable. Ciudad de origen del vuelo.
- *destino*, de tipo String, consultable. Ciudad de destino del vuelo.
- *fecha de salida*, de tipo LocalDateTime, consultable y modificable. Fecha y hora de salida del vuelo.
- *número de plazas*, de tipo Integer, consultable. Número de plazas del vuelo.
- *número de pasajeros*, de tipo Integer, consultable y modificable. Número de pasajeros del vuelo.
- *completo*, de tipo Boolean, consultable. Indica si el vuelo está completo, es decir, no quedan plazas libres.
- *cadena con formato*, de tipo String, consultable. El código de vuelo entre paréntesis, seguido del origen y el destino separados por un guión y la fecha y hora de salida. Por ejemplo: "(IB2089) Sevilla – Madrid 25/12/2016 17:25"

Constructores:
- C1: recibe un parámetro por cada propiedad básica del tipo.
- C2: recibe los mismos parámetros salvo el número de pasajeros, que se inicializa a 0.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

6. Beca

Paquetes: fp.tipos.universidad, fp.tipos.universidad.test

Propiedades:
- *codigo,* de tipo String, consultable. Código de la beca.
- *cuantiaTotal*, de tipo Double. consultable. Cuantía total de la beca.
- *duracion*, de tipo Integer, consultable y modificable. Número de meses que dura la beca.

- *tipo*: de tipo TipoBeca, consultable. Puede tomar los valores ORDINARIA, MOVILIDAD, EMPRESA. Representa el tipo de beca.
- *cuantia mensual*: de tipo Double, consultable. La cuantía mensual se calcula como la cuantía total entre el número de meses que dura la beca.

Constructores:

- C1: un parámetro por cada propiedad básica del tipo.
- C2: permite crear becas de las que no se conoce aún su cuantía ni su duración. Por defecto la cuantía será de 1500,0 € (definirlo como constante), y tendrá una duración de un mes.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.

7. Persona (fp.tipos.common)

Paquetes: fp.tipos.common, fp.tipos.common.test

Propiedades:

- *nombre,* de tipo String, consultable. Nombre de la persona.
- *apellidos,* de tipo String, consultable. Apellidos de la persona.
- *dni*, de tipo String, consultable. DNI de la persona.
- *fechaNacimiente,* de tipo LocalDate, consultable. Fecha de nacimiento de la persona.
- *edad:* de tipo Integer. La edad se calcula a partir de la fecha de nacimiento.

Constructores:

- Un constructor con un parámetro por cada propiedad básica.

Representación como cadena: una cadena con el nombre de la clase y todas las propiedades del tipo separadas por comas.